AU-A-80579/98

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 | A1

(11) International Publication Number: WO 98/57279

(43) International Publication Date: 17 December 1998 (17.12.98)

(21) International Application Number: PCT/US98/11493

(22) International Filing Date: 12 June 1998 (12.06.98)

(30) Priority Data:
60/049,623 13 June 1997 (13.06.97) US

(71) Applicant *(for all designated States except US)*: UNIVERSITY OF MARYLAND [US/US]; Office of Technology Liaison, 4312 Knox Road, College Park, MD 20742 (US).

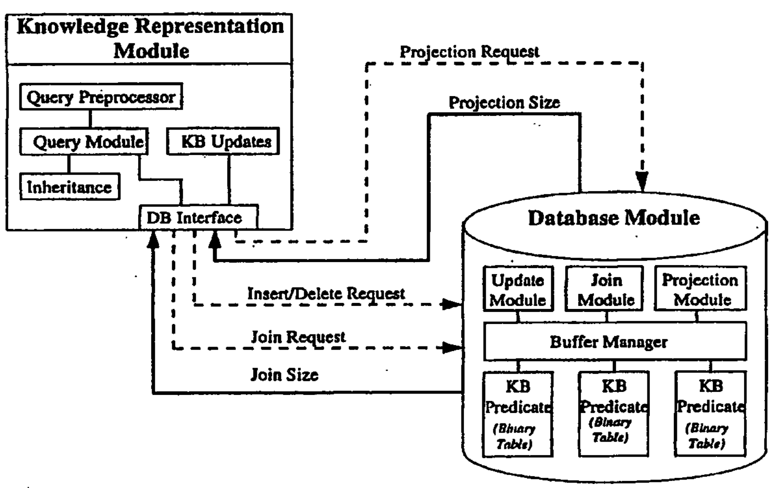
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: HENDLER, James, A. [US/US]; 14007 Arctic Avenue, Rockville, MD 20853 (US). STOFFEL, Kilian [CH/CH]; Ruedes Vermes 11-B, CH-2103 Colombier (CH). TAYLOR, Merwyn, G. [US/US]; 11252 Evans Trail #202, Beltsville, MD 20705 (US).

(74) Agent: GOLDHUSH, Douglas, H.; Nikaido, Marmelstein, Murray & Oram LLP, Metropolitan Square, Suite 330, 655 Fifteenth Street, N.W., Washington, DC 20005-5701 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: KNOWLEDGE REPRESENTATION SYSTEM INCLUDING INTEGRATED KNOWLEDGE-BASE AND DATABASE, AND METHOD AND APPARATUS FOR UTILIZING THE SAME



(57) Abstract

A knowledge representation system includes a database module for storing data in a database format, and a knowledge representation module for storing knowledge information in a knowledge-base format. The knowledge representation module includes an interface mechanism for interfacing communication between the database module and the knowledge representation module. A query processing device is provided for processing queries input into the knowledge representation system. The query processing device initializes communication between the knowledge representation module and the database module based upon preprocessing of queries. A single query to the query processing device results in communication between the database module and the knowledge representation module through the interface mechanism.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

TITLE OF THE INVENTION:

KNOWLEDGE REPRESENTATION SYSTEM INCLUDING INTEGRATED KNOWLEDGE-BASE AND DATABASE, AND METHOD AND APPARATUS FOR UTILIZING THE SAME

BACKGROUND OF THE INVENTION:

Field of the Invention:

The present invention relates to a method and apparatus for providing a highly efficient knowledge representation system which utilizes what have been referred to as knowledge-base technologies and relational database technologies. Recent research in knowledge-base systems has concentrated on systems which are capable of storing significant amounts of data in implicit formats, and accessing this data in an efficient manner. Recent developments provide high-speed query results for large-scale complex "implicit" knowledge bases only when utilizing expensive computing systems with parallel processors having significant amounts of random access memory. Conventional databases, which typically only use explicit data fields, can provide fast results with less complicated computers; however, the query results are based only upon searches of these explicit data fields. The present invention, therefore, seeks to integrate differing technologies to provide an efficient large-scale knowledge representation system which can function using what would currently be considered to be an inexpensive or typical single processor personal computer (PC).

Description of the Related Art:

Traditional relational database management systems (RDBMS) perform what is known as "explicit" data retrieval by explicitly matching data fields in a query or search request with data fields stored in the database. At the time of the creation of the database, or entry of data into the database, data is entered in a particular field. For example, a medical database may contain fields for "patient name", "patient address", "date-of-birth", "sex", etc.

More complex systems are referred to as knowledge bases or knowledge representation systems, instead of databases. Knowledge bases can utilize what are known as implicit relationships between seemingly unrelated or distantly related elements in order to provide query results based upon this implicit knowledge, rather than by merely matching explicit data fields or relationships. Such implicit search capabilities have often used what is known as a semantic network system, and is discussed at length in a paper by Evett, Hendler, and Spector, entitled "Parallel Knowledge Representation on the Connection Machine" (Journal of Parallel and Distributed Computing, vol. 22, pages 168-184, Academic Press, Inc. 1994). The content of this paper is hereby incorporated by reference. As explained in this paper, semantic networks have been known to be useful in knowledge representation, and are commonly used for contemporary artificial intelligence research. However, semantic networks require a significant amount of data in order to properly interrelate a series of elements. For illustrative purposes, Figure 1 illustrates a small portion of what could be a large semantic network. The objects of the network, also called nodes or frames, are named within the ovals. A common link in semantic networks is the "Is-a", or "ISA", to connect nodes representing classes of objects to those nodes representing subclasses or superclasses. The subgraph consisting completely of ISA links, and the nodes utilizing them, is referred to as the ISA hierarchy of the semantic network. Conventional nomenclatures such as parent, child, ancestor, and descendant are used to refer to nodes representing superclasses, subclasses, etc. The illustration in Figure 1 is a semantic network which represents some information from a hypothetical marketing application -- information about people, their ages, their education level and their vehicles. Dark lines represent ISA links, while light lines, with labels, represent properties of the various frames. From this diagram it can be seen, for example, that cars separate into the categories of domestic cars vs. foreign cars, that people can be computer scientists or pathologists, etc. In addition, property links show the ages of various people, the degrees

associated with various professions, and which people drive which kinds of cars. Such semantic networks typically utilize an inheritance mechanism in cooperation with the ISA hierarchy. Property inheritance is typically used to minimize the number of property links in the network. For example, it is possible to indicate, in a network regarding animals, that mammals are warm blooded by placing a blood temperature link between a "warm blooded" node and each node representing an instance of a mammal. A different solution can be provided utilizing property inheritance by placing a single blood-temperature link between a mammal node representing the class of all mammals, and warm-blooded. Any descendants of the mammal node would therefore inherit the warm-blooded property of the blood-temperature from their ancestor. In the knowledge base discussed by Evett, Hendler, and Spector, nodes are referred to as explicitly valued for a given property if the node is incident on a property link of the given property. Any node which is not explicitly valued inherits the value of its nearest ancestor (nearness as determined by ISA links) which is explicitly valued for that property. In situations of multiple inheritance, the inherited value is considered to be a function of the property values of those nodes. Through the use of an inference mechanism called activation wave propagation, a set of nodes is activated according to a parallel type of search referred to as a breadth-first graph search. The activation wave propagates synchronously, in that all nodes in the current wave front activate incident nodes which have not yet been activated. When all incident nodes are activated, thereby forming a new wave front, the wave is continuously propagated. As the activation wave propagates along the ISA links, synchronous, parallel propagation occurs across the wave front. As the knowledge base becomes large, tremendous computing power is necessary to perform this propagation in parallel. However, because of the relationship of inheritance in the ISA hierarchy, two different types of queries and inferencing are performed. Bottom-up queries are more specific queries such as "what color is an elephant"; top-down queries, on the other hand, are much more indefinite, such as "what things

are gray." Using Figure 1 as an example, answering the question "What degree does Jane have?" relies upon bottom-up inferencing; a parallel knowledge and association machine would create an activation wave with "Jane" as the sole element, and would then work from the bottom up. Another example of property inheritance is what is known as top-down inferencing; an example of a top-down query would be "what objects are people?" The system would then make the node "person" the first element in the activation wave, and since it has children in the ISA hierarchy, would propagate the wave downward until no new nodes were activated. The time necessary for complete propagation of a wave through a network is proportional to the depth of the network. Typically, bottom-up inferences in large topologies utilize a simple pointer chase with numerous address and directions, serial computing methods can be used with an acceptable response time. Top-down inferencing, however, makes the use of serial technology prohibitively time consuming. Top-down calculations typically require traversal of large portions of a semantic network, and moving downward the time necessary is based upon the number of nodes, rather than the depth of the network. Answering a simple top-down query in a very large semantic network could take several minutes on a serial computer. Large network serial implementations, therefore, can run only bottom-up or in very limited top-down inferences. Although various computing environments enable differing time-scales based upon particular applications or users needs, computational effectiveness typically seeks a minimum amount of time for answering queries. Evett, Hendler, and Spector utilized a Connection Machine, which is a single instruction multiple data (SIMD) parallel machine wherein a plurality of microprocessors operate in parallel; the knowledge base is implemented in software, and was written in "* LISP" (star lisp), where appropriate controllers are used to take advantage of the parallel nature of the connection machine. A series of parallel variables, or pvars, are used to control parallel operations of the plurality processors. Parallel processing in accordance with this paper, however, requires, by virtue of the propagation of the activation

wave, very significant amounts of memory; virtually all of the data of the semantic network was required to be stored in internal memory. A similar technique was also used for more generic multiple instruction multiple data (MIMD) supercomputers. Typically, all such computers are extremely expensive, on the order of several hundreds of thousands of dollars or more in 1998 dollars.

SUMMARY OF THE INVENTION:

The present invention is directed to an integrated knowledge-base and database technology which utilizes a high-performance implementation of a knowledge and association system in combination with a relational DBMS in order to efficiently support complex queries on extremely large knowledge bases using a single processor, rather than requiring massively parallel systems, as required in the prior art. A knowledge and association system according to the present invention can operate on a computer system having less than 16 MB of random access memory (RAM) to hold a knowledge base of over two million assertions. The configurations of the present invention therefore provides a significant reduction in the need for storage, with little effect on calculation time. The load time of very large knowledge bases can therefore be reduced by more than two orders of magnitude. Processing time on a single processor can also significantly outperform previous supercomputer implementations.

BRIEF DESCRIPTION OF THE DRAWINGS:

For a proper understanding of the present invention, reference should be made to the accompanying drawings, wherein:

Figure 1 illustrates a portion of a semantic network, including objects and links;

Figure 2 illustrates communication between a KR module and an RDBMS module according to the present invention;

Figure 3 illustrates three layers of an embodiment of the present invention;

Figure 4 illustrates a left deep join tree for query evaluation according to the present invention;

Figure 5 is a block diagram of a hardware configuration according to the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS:

The present invention is a knowledge representation system which is based upon a relational database system. The knowledge representation system is tightly coupled with a RDBMS system such that bigger knowledge bases can be accessed, and the system's inference engine effectively and efficiently utilizes system memory.

The invention is a frame-based artificial intelligence system, utilizing a property/class system, which is efficiently supported using scalable computing techniques. The invention enables scaling to extremely large size applications, which has become important based upon the current boom in information technology. The knowledge representation system of the present invention allows the user to define a frame-based knowledge base with class, subclass, and property links to encode the knowledge base. The term "ontology" is sometimes used to refer to such knowledge bases. Although numerous interpretations of the term "ontology" currently exist, for the purposes of this invention the term ontology should be considered to be synonymous with the term "knowledge base". Property values of the knowledge base can be frames, strings, numeric values, or specialized data structures. Exceptions to the various links can be made in the forms of multiple-inheritance, and inheritance can be efficiently performed utilizing a true inferential-distance-ordering calculation of a semantic network. Complex structure matching queries, which are conjunctive queries relating to variables and constraints, can be processed. While prior art systems enabled these advantages through the use of massively parallel computers or supercomputers, the present invention is capable of supporting the demands of extremely large knowledge bases, without requiring the use of such computers. This significant advantage is provided by utilizing an increasing

number of database algorithms to perform inferencing. By configuring a system which can be compared to the previously known Parka-KR (tm) system using a relational database management system (RDBMS), the invention is capable of blending the knowledge-base inferencing capabilities of the prior art knowledge and association system with the standard database capabilities of RDBMS. The present invention uses the RDBMS as a run time storage medium. Due to the fact that the RDBMS uses external devices to store data, the invention is capable of managing knowledge bases which are too large to be maintained in internal memory. The present invention can therefore handle knowledge bases which are as large as available disk space, and make use of the RDBMS to efficiently manage the input and output between the primary and secondary storage. The invention, therefore, is capable of scaling arbitrarily large knowledge bases while outperforming other knowledge representation systems.

In order to efficiently inference concepts in the invention, concepts are differentiated into two categories; structural assertions and non-structural assertions. Structural assertions consist of relations such as ISA, INSTANCE, INSTANCE OF, and SUBCAT, which encode the class/subclass ontology in a parallel knowledge and association knowledge base. Non-structural assertions, therefore, are all other concepts in the knowledge base. Such a differentiation is important, structural assertions are used in property inheritance. This is critical in the present invention, due to the fact that the invention scans the structural assertion for computing inherited properties, and the inheritance evaluation is a critical aspect of all queries issued to the system. Therefore, structural assertions are stored in internal memory, for high speed access. The number of structural assertions is typically much less than the number of non-structural assertions. Therefore, structural assertions can be stored in internal memory even in the largest known knowledge bases.

The non-structural links for knowledge bases, on the other hand, typically require more memory than is currently available on all but the largest and most expensive computer systems, such as supercomputers. Due to the

fact that supercomputers are extremely expensive, the amount of internal memory which can be found in supercomputers cannot be compared to the amount of internal memory which can be found in more affordable computer systems. Additionally, it is highly unlikely that all data contained in a set of non-structural assertions will need to be processed for any given query. This is significantly different from structural assertions, which are often all scanned for a given query. For this reason, it is not necessary to store non-structural assertions in primary or internal memory at all times. The present invention, therefore, is configured such that non-structural assertions are stored in specialized tables within the database, which allows the database system to handle loading of only the subset of non-structural assertions required to elevate a given query. This is a process which RDBMS systems can efficiently manage. The present invention, therefore, can access the highly utilized structural assertions in a rapid and efficient manner, and rely upon the database system to manage the larger and less frequently accessed individual items of non-structural data efficiently.

Referring to Figure 5, a computer configuration is generally illustrated wherein CPU 10 receives input from input device 20. Display 30 is used to display input and output from CPU 10. Random access memory (RAM) 40 and disk storage device 50 are connected to the CPU. RAM 40 is sometimes referred to as primary memory, and disk storage 50 is sometimes referred to as secondary memory. When a knowledge representation system according to the invention is operating in conjunction with CPU 10, queries are input via input device 20, and fed into CPU 10. CPU 10 then parses the query to identify properties, constants, and variables, and generates a query evaluation plan. The query evaluation plan includes, based upon instructions in RAM 40 and disk storage 50, with the query evaluation plan including the searching of structural assertions regarding property inheritance. The invention also allows the structural tables to be divided into subtables and paged to memory; however, this is rarely needed in practice. The knowledge representation system according to the present invention is such that all

structural assertions regarding property inheritance are usually stored in RAM 40. This enables high speed access to the structural assertions. Once structural assertions are scanned, it is then possible to scan disk storage 50 for appropriate non-structural assertions. The present invention stores, as noted previously, non-structural assertions in specialized tables within disk storage 50; this enables loading of only a subset of non-structural assertions, as may be necessary for a given query, into RAM 40.

As shown in Figure 2, interaction between the knowledge representation module and the database module according to the invention can be illustrated by the interactions which occur when a query is being answered. A projection request is made from the non-structural assertion table, and a request is made to perform a join over two tables. Utilizing this configuration, the processing power of the database for those services which can be performed by the RDBMS can be efficiently performed; the services are primarily directed to join processing. The knowledge representation component of the invention relies on the database not only for the storage and retrieval services, but also for the intermediate processing of complex queries. The invention, therefore, can take advantage of database optimizations which may be present.

The present invention can be implemented on top of numerous types of RDBMS systems. In one implementation, the invention is implemented on top of, for example, an RDBMS system which is known as Parka-REL (tm). Parka-REL offers standard DDL interface for creating databases. Relational tables can be created in a straightforward manner, through direct function calls from a C program, or through a parser. In order to provide efficient access to user data, Parka-REL provides users with the opportunity to create indexes on attributes of relational tables. Once the tables have been defined and created, users can seamlessly insert and delete records. Parka-REL is an example of a system which provides storage and retrieval functionality, and all of the standard database operations required in the framework of an integrated knowledge-base and database system according to the present

invention. The invention communicates with the RDBMS system directly through the systems library of C function calls, rather than through SQL (structured query language), although a limited SQL interface is provided for user access to the database. This particular RDBMS, though more limited in functionality than some other systems, is very useful and supports a configuration according to the present invention in an adequate manner. The limitations of a system such as Parka-REL are more than outweighed by the fact that Parka-REL may become easily accessible.

An important aspect of the present invention is that although the invention is a frame-based knowledge representation (KR) system, the knowledge base is stored as a series of relational tables, in order to take full advantage of the processing services of the RDBMS. Each assertion type is associated with a relational table in the RDBMS. Frames in the present invention, therefore, have their information distributed among a number of tables.

When evaluating complex queries on large knowledge bases, the size of intermediate results can be significant. Computing the results of such queries utilizing only internal memory can place a significant demand on system resources, especially if the data to be processed is also stored in internal memory, which is the case in most knowledge representation systems. The present invention overcomes this problem by distributing the workload such that the knowledge representation component prepares relational tables for the RDBMS component to process, using external storage as needed. The RDBMS unit, therefore, can consume available disk space while processing intermediate results, thereby reducing the demand for valuable internal system resources.

By keeping the inheritance structure or structural assertions in primary memory, the largest knowledge base used in development of the invention required only 1.1 M bytes of RAM. Additional memory was occupied by the Parka-REL RDBMS; however, only 40 buffers of 4K bytes each were required; utilizing a larger buffer pool or bigger page size would allow the

RDBMS to store more data in physical memory, thereby reducing the amount of disk I/O. However, in the tested configurations, the resources consumed by the graphical query interface, the RDBMS, and the present invention never exceeded 16M bytes of physical memory. As can be seen in Table 1, a comparison is shown of the loading times for a prior art system and the present invention, utilizing knowledge bases of three different sizes. Knowledge bases known as Caper-20, Caper-100, and Caper-200 are used, having differing numbers of frames, structural links, and assertions as shown in the table. The knowledge bases were generated for Case based planning work in accordance with the present invention, and were created via a generative planning system. Information is contained in each of these knowledge bases on the plans, sub-plan structure, and causal information used by the planner, as well as ontology information about a logistics-planning domain. The biggest knowledge base, Caper-200, has over two million assertions, and can be loaded in under three seconds utilizing the present invention. This, as shown in the table, is over 200 times faster than the prior art; this significant advantage is provided due to the fact, according to the present invention, that the knowledge base is saved in a significantly denser format, which reduces the number of input/output operations. As discussed above, only a small part of the knowledge base, that being the structural assertions, is actually loaded into internal memory.

Table 1

Knowledge Base	# Frames	# struct. Links	# Asserts	Prior Art	Present Invention
Caper-20	11103	26404	118615	28	0.29
Caper-100	56516	130526	807480	214	1.34
Caper-200	116297	266100	1635782	640	2.80

The use of secondary memory such as disk space can result in a loss of efficiency, due to the result of the trade-off between primary memory and time to load information. Despite this, the system remains extremely efficient due

to the use of database technology to manage this trade-off. As shown in Table 2, several queries were performed on the largest of the Caper knowledge bases described previously, and analyzed for performance. Q1 is a query that finds planning variables and their corresponding values, it uses one variable and 2 predicates. Q2 is a query to find all cases where a vehicle was moved by train through a particular hub, it uses 5 variables and 7 predicates. Q3 is a query to find all cases where a regular truck was used to move a package, it uses 6 variables and 7 predicates. Q4 is a more complex query concerning 8 variables, 8 predicates over 17 network links. It asks to find all cases where a liquid was moved through a city location using any kind of tanker truck. The times shown are in milliseconds, and the queries were run on a single processor Sparc Ultra 1 with 64M of RAM. As can be see, query times stay low and the time spent in inferencing (KB and DB times) dominates over the I/O times.

A more conceptual view of the present invention can be found in Figure 3. Figure 3 illustrates the three major components of the invention, with the first component being the application interface or API layer. This layer is used to create, alter, and query knowledge bases. The inferencing layer is the middle layer of the inventive architecture, and manages concept taxonomies, and performs hierarchical inferencing. All requests for super-categories and sub-categories of concepts are serviced at this layer. This layer is configured to contain an efficient data structure for storing hierarchical links that provide dynamic access to hierarchical information about frames in the knowledge-base of the invention. The inference layer contains efficient inheritance, hierarchical traversal, and look-up algorithms. The RDBMS layer is discussed above, and is configured with the inference layer with a sufficiently tight coupling such that access is provided to low-level objects and procedure calls.

Table 2

Caper 200	Query Size	# matches	KB time	DB time	I/O time	Total (msec)
Q1	2 var, 2 pred	6221	12	111	49	162

Q2	5 var, 7 pred	101	50	188	124	362
Q3	6 var, 7 pred	269	68	258	92	418
Q4	8 var, 8 pred	23	157	686	270	1113

As noted previously, the present invention stores data in relational tables. The structural data is stored in a three-column relational table, and the non-structural data is stored in individual binary relational tables, with one table being provided for each non-structural property. When a knowledge-base is opened for the first time, the invention reads the relational table containing the structural assertions into memory. As noted previously, this data is loaded into memory due to the fact that the data is always accessed during query evaluation. The inheritance algorithms supported by the invention access the structural data to determine which frames will inherit properties from their ancestors. The structural data is used to generalize values at lower levels in the taxonomy to those that are at higher levels. Storing the structural data in memory reduces the time necessary to access the data, and therefore improves the overall query evaluation performance. The structural data is stored as an array of four variable length vectors. The vectors are used to store the immediate descendants and ancestors of frames. This data structure is ideal for traversing concept taxonomies, one level at a time. Additionally, the present invention is capable of retrieving all ancestors of a given leaf-level concept in an efficient manner. The present invention, by virtue of the tight coupling between inferencing and RDBMS, enables the timely response of the present invention to request for ancestors of leaf-level values, and for the inheritance of property links as described above.

The use of a binary table representation for non-structural properties is advantageous in that the inheritance models are easier to implement on single properties. Frames that inherent properties from other frames will not consume space. Additionally, the use of relational tables to represent properties avoids the disk input/output overhead of loading all properties of frames into memory to gain access to only a few of them. This is a primary

difference between this invention and what are known as object-oriented databases (OODB). OODBs store all information about an item in contiguous memory locations, and bring all this information into memory when queried. The present invention utilizes relational tables and distributes the information about a particular entity among many such tables to gain efficiency.

The evaluation of queries according the present invention is based on expression of queries in a Lisp like syntax. The following is an example of a query using Figure 1:

(Drives ?per ?car)(isa ?car Foreign Car)(Degree ?per ?deg)

This is a request for all tuples having a "drives" attribute which is equal to a value which is a descendant of the type "Foreign car." The query also asks for the corresponding class value of Degree for each of the values returned by the first two clauses. This query references relational data and data from the concept taxonomies. The two clauses "(drives ?per ?car)" and "(Degree ?per ?deg)" are references to particular database tables, while the other clause "(isa ?car foreign car)" is a reference to the class/subclass relationship with respect to cars. This query illustrates the ability of the invention to merge concept taxonomies (the ISA hierarchy) with relational information.

The invention evaluates queries in three phases; parsing, generation of an evaluation plan, and execution of that plan. Queries are parsed, to identify all properties, constants, and variables. After parsing, a query evaluation plan is generated, which is typically represented as what is referred to as a left deep tree. A natural join is performed on tuples that satisfy the individual subexpressions. All queries are conjunctions of subexpressions expressed as triplets in the form (prop domain range). The left deep query tree represents the join order for the "domain" and the "range" variables. Heuristics are used to generate the join trees. The query plan is then executed by performing a series of selection, inheritance processing, and joining for all subexpressions. The order in which subexpressions are inserted into the query plan effects the time that is spent evaluation the query.

A query tree is initially seeded with the subexpression that will theoretically have the smallest result set. Determining which subexpression will have the smaller result set is complex because the result sets can grow in size after property value inheritance. Although some commercial databases maintain data distribution statistics for each table, with these statistics determining joint orders that will minimize intermediate result size, this strategy cannot be applied according the present invention, because data distribution statistics cannot measure the effects of inheritance without extensive preprocessing. Queries are thus commonly represented as directed graphs. The constants are variables are represented as nodes and the predicates/properties are represented as links. A strongly connected component is defined as a sub-graph whos links, L, satisfy at least one of the following constraints:

1. L is a user defined property OR
2. L is a predefined property AND the source or the destination node of L is a constant.

Links that connect strongly connected components are predefined predicates that have two unbound constraints, ie the source and destination nodes are variables. Therefore, a query can be represented as a set of strongly connected components which are in turn connected by links in the graph. Optimizing the evaluation of a query is reduced to optimizing the evaluation of the strongly connected components and recombining the components based on the links that connect them.

The left deep tree for a connected component is created using heuristics that determine the order in which to join the triplets contained in a connected component. The order in which triplets are inserted into the tree effects the time that is spent evaluating the connected component. The heuristics are used to minimize this time. A left deep tree is initially seeded with the triplet that will theoretically have the smallest result set. This is usually a triplet containing a structural predicate and one constant as a constraint. If a connected component contains a set of triplets that have constants for both constraints, the triplets are scheduled for evaluation first

because their result sizes are zero. However, they are not added to the join tree. A tree is expanded by adding triplets which contain variables or constants that are present in nodes that have previously been added to the tree. In other words, triplets are added for which a natural join with the intermediate tree is possible. Triplets are added one at a time to the left deep tree. If there exists more than one candidate to add to a tree, the triplet that joins with two variables is preferred. If there exists two or more such triplets, one is chosen at random. In situations where there exists two or more candidates to add to a tree and all candidates join with a single variable in the tree, the candidate that has the theoretically smallest result set is selected.

Special operators such as "StringMatch", "ge"(greater than), and "le" (less than) that have one constant and one variable as constraints are scheduled for evaluation after the corresponding "variable" has been added to the left deep join tree. These operators are evaluated as filters on variables as opposed to triplets which are joined into the left deep join trees. The evaluation is performed after the variable is instantiated. For the query described earlier

(Drives ?per ?car)(isa ?car Foreign Car)(Degree ?per ?deg)

a left deep join tree is generated as shown in figure 4. The query evaluation plan is generated as follows:

1. Find all leaf-level descendants of "Foreign Car" and store in intermediate table I1(?car)
2. Select all person and car values from the "drives" relational table and store in I2(?per ?car). Perform a natural join between I1(?car) and I2(?per ?car) and store results in I3(?car ?per).
3. Select all person and degree values from the "degree" relational table and store results in temporary table T1. Calculate inheritance (according to algorithm described below) and store in temporary table T2. T2 includes the inherited values (T will contain the tuples <Jane, PhD>,<Bob, PhD>, <Bob,MD> and <Bill, MD> with the two values for Bob resulting from his

inclusion in both the computer scientist and pathologist classes.) Compute I4 as the (relational) union between T1 and T2.

4. Join I3 and I4 on the column ?per and store in Final(?car ?per ?deg). This plan begins with computing all descendants of Foreign Car. This was selected because it results in a single column intermediate table with few values. The second subexpression (drives ?per ?car) was selected because it is the only expression that joins with I1(?car) using the ?car variable. Finally, (Degree ?per ?deg) is added to the join tree because the ?per variable joins with the ?per variable in the intermediate table I3(?car ?per). The final table FINAL(?car ?per ?deg) contains the results of the query. The contents of FINAL are listed in Table 3.

Table 3

?car	?per	?deg
BMW	Bob	PhD
BMW	Bob	MD
Honda	Bill	MD

To further clarify the invention, it may be appropriate to discuss, in more detail, the implementation of the inferential distance ordering, linking, and analysis which is performed by the present invention.

While ISA links directly encode class and type hierarchy information in a knowledge representation system according to the invention, their primary purpose is to provide for property inheritance. Any frame that is not explicitly labeled for a given attribute inherits its value from a predecessor that is explicitly labeled for that attribute. In effect, attribute values "flow downward" across ISA links. The invention's inheritance mechanism is therefore "top-down". Exception handling is the problem of handling situations where a node has more than one explicitly valued predecessor. The invention utilizes, as noted previously, an inferential distance ordering (IDO) protocol as one

example of a way to choose among such predecessors, selecting the "nearest" one, as defined by the IDO. If there are more than one such predecessor (which is possible because the invention allows multiple inheritance), the frame does not have a value for the attribute.

A fundamental assumption behind the way the IDO handles inheritance exceptions is: subclasses override superclasses. Assume the object in question, X, is not explicitly labeled for the given property, P (if it is, the explicit value is X's value for P). Let B be the set of predecessors (B_1, B_2,...B_l) of X that are explicitly valued for P. X takes B_i's value for P as its own, provided B_i is an element of B such that there is no B_k ($k \neq i$), that is also a successor of B_i. If more than one element of B meets this criterion, X is said to be ambiguously valued for property P. If B is empty, X is not valued for property P. IDO is one mechanism for resolving ambiguously valued properties.

The basic inheritance algorithm determines the values of a given attribute for every frame in the KB. The invention uses a "marker-passing" technique to effect a traversal of the KB's inheritance graph. The basic idea is that markers cascade downward across the successor arcs of the network, from general to more specific nodes. Markers originate at the nodes that are explicitly labeled for the attribute in question. Each marker consists of a pointer to the node where it originated (we call this pointer the "ID"), and a polarity (positive or negative). Markers are written as "-ID" or "+ID". A positive or negative copy of a marker originating at a node will reach all its successors. Markers become negative when propagated through an explicitly labeled node. In general, at the end of the algorithm, if a (non-explicitly-labeled) node receives only a positive marker for a certain ID, the node inherits the attribute value of the predecessor corresponding to the ID. If, however, the node receives both a positive and negative marker of the same ID, it does not inherit from that predecessor.

When markers originate, they have a positive polarity. Upon reaching an explicitly-labeled successor, the markers obtain a negative polarity. As

negative markers cascade downward through the successor arcs, they "cancel" any corresponding positive markers that they reach. The intention of this design is to effect a mechanism that implements appropriate relations for the chosen IDO. Note that the addition of any number of non-explicitly labeled frames along either path would not change the results of the algorithm. Summarizing the algorithm, the following steps are used for computing property inheritance:

1. **Initialize**. Identify all the frames that are explicitly labeled for the given attribute. These frames form the starting point for the traversal algorithm, and initially comprise the wave front. Each frame in the wave front maintains a list, outgoing, of markers to be propagated and initializes outgoing to consist of a single positive marker with its own ID. Each frame also initializes a list, $list_{sent}$ in the same way.
2. **Propagate**. The members of the wave front now activate all their direct successor frames by sending them a copy of the markers in outgoing, simultaneously placing. The frames receiving these markers become the new wave front, placing incoming markers in a list, incoming.
3. **Filter**. The (new) wave front frames now determine if they are explicitly labeled for the attribute.
   - A. For all frames that are explicitly labeled, in the next iteration they will propagate to their successors a negative version of the markers just received, in incoming. This will tell their successors that if they receive markers with these Ids, they are no longer valid. They do this by placing a negative version of each marker in incoming into outgoing, and into $list_{sent}$.
   - B. If a frame is not explicitly labeled, it performs the following operations:

a). For each marker m in incoming it checks if a positive of negative version of the marker is already in $list_{sent}$.

b). If the no such marker is found in the list, m is inserted into $list_{sent}$ and outgoing.

c). If m is found in $list_{sent}$ with the same sign, it can be ignored for all further operations; the marker m was processed and send to all successor nodes in a previous iteration.

d). If m is found in $list_{sent}$ with a different sign, the following situations must be analyzed:

e). If the incoming marker has a negative sign, -ID, (and so there is positive copy, +ID, in $list_{sent}$), the incoming (negative) marker must be propagated to all successors, and so is placed in outgoing. Also, +ID is removed from $list_{sent}$ and replaced with the -ID. The negative marker is said to cancel its positive counterpart.

f). If the incoming marker has a positive sign, +ID, it can be ignored. The traversal algorithm stops when there are no more active messages, i.e. when all messages have reached a leaf node in the DAG or have been cancelled during the traversal.

After the traversal terminates, each frame analyzes its $list_{sent}$. Each positive marker represents that a value for the given attribute is inherited from the frame with the corresponding ID. If a frame has more than one positive marker in its $list_{sent}$, a conflict exists in the inheritance hierarchy that cannot be resolved by the IDO mechanism.

To obtain the final values for the given attribute for each frame, the following rule is applied: explicitly labeled frames maintain their local attribute

value, while non-explicitly labeled frames inherit the value from the frame whose ID is the only positive marker in $list_{sent}$. By assigning all frames their inherited values the algorithm is terminated. The same algorithm can be used to compute other inheritance types than inferential distance ordering. "Credulous" inheritance, returning all possible values, can be computed by returning all values for ambiguous frames. Other forms of inheritance can be performed by either disallowing the propagation of negative markers, or by not removing +ID in step (e). The invention provides for a parameter to be specified which controls which of these techniques will be used.

The semantic network, as noted previously, is stored in two types of relational tables. The properties that appear as links in a semantic network are divided into two categories, structural links and non-structural links.

The structural assertions are stored in a single tertiary relational table. The scheme for this table is structural (property-id, source-id, destination-id) where property-id is the frame identifier for any member of the set {"isa", "subCat", "instance", "instanceOf"}, source-id is the frame identifier of the frame for which the property is being asserted, and destination-id is the frame identifier of the frame that is the object of the assertion. The structural assertions are loaded into memory when a knowledge base is opened. To reduce the start-up cost of loading the structural assertions, another version of the structural assertions is stored in a binary file. The file is a list of structural links. For each frame in a knowledge base, there exist four lists, one for each type of structural assertion.

The non-structural assertions are stored in binary tables, one binary table for each assertion name. A non-structural assertion is defined as any assertion that is not in the set {"isa", "subCat", "instance", "instanceOf"}. The scheme for non-structural assertions is property (source-id, destination-id), where property is the name of the assertion being made, source-id is the frame identifier of the frame for which the property is being asserted, and destination-id is the frame identifier of the frame that is the object of the

assertion. Non-binary relations are handled using tables corresponding to the relation size.

Tables 4, 5, and 6 are relational table representations of a semantic network regarding names, and subcategories and supercategories.

Table 4

Structural link	Source	Destination
Instance	Jane	Computer Scientist
Instance	Bob	Pathologist
Instance	Bill	Pathologist
Isa	Pathologist	Person
Isa	Computer Scientist	Person
Instance	Ford	Domestic Car
Instance	BMW	Foreign Car
Instance	Honda	Foreign Car
Isa	Foreign Car	Car
Isa	Domestic Car	Car
Isa	Person	Thing
Isa	Car	Thing

Table 5

Degree	
Computer Scientist	PhD
Pathologist	MD

Table 6

Drives	
Jane	Ford
Bob	BMW
Bill	Honda

Table 7

Age	
Jane	35
Bob	49
Bill	51

As noted previously, a knowledge representation system according to the present invention enables efficient utilization of computing capabilities, while providing very efficient and rapid query processing of very large knowledge bases. Through the tight integration of knowledge-base and database technologies, and the effective and novel use relational tables, complex queries can be performed utilizing a single processor, where massively parallel systems were formerly required. It should be noted, however, that while the present invention enables effective use of complex queries utilizing a single processor, the same types of advantages with respect to load time and accessing speed would result from the use of the invention on a multiple processor system.

It should be noted that the above-discussed embodiment is discussed to the utilization of a knowledge representation system in conjunction with an RDBMS layer. It should be noted that the critical and unobvious advantages of the present invention can be realized through the utilization of query processing and inheritance of the present invention, in conjunction with various types of databases, such as commercially available text retrieval systems, global positioning systems (GPS), language translation systems, etc., as available from numerous sources.

The present invention can be embodied as a computer program embodied on a computer readable medium, which is loaded on or stored on a general purpose computer. The invention can also be embodied as a method of processing queries, with the method comprising a series of steps regarding integration of knowledge representation and database technologies. The invention may also take the form of a series of hardware modules, hardware elements, or other devices which are configured to perform the

necessary functions to provide the critical and unobvious advantages discussed above.

The above descriptions of the invention are for illustrative purposes only, and do not limit the scope of the invention. Numerous modifications can be made to the descriptions set forth above, while still remaining within the spirit and scope of the invention. For proper determination of the metes and bounds of the present invention, reference should be made to the appended claims.

CLAIMS:

1. A knowledge representation system, comprising:

a database module for storing data in a database format;

a knowledge representation module for storing knowledge information in a knowledge-base format, said knowledge representation module including an interface mechanism for interfacing communication between the database module and the knowledge representation module;

a query processing device for processing queries input into the knowledge representation system, said query processing device initializing communication between the knowledge representation module and the database module based upon preprocessing of queries, whereby a single query to the query processing device results in communication between the database module and the knowledge representation module through the interface mechanism.

2. A knowledge representation system as recited in claim 1, wherein said database module contains a plurality of relational tables, said relational tables storing structural inheritance data used by the knowledge representation module.

3. A knowledge representation system as recited in claim 2, wherein the interface mechanism is configured to coordinate communication between relational tables in the database module and the knowledge representation module.

4. A method of processing queries in a knowledge representation system, said method comprising the steps of:

inputting a query to the knowledge representation system;

parsing the query to identify properties, constants, and variables contained therein;

generating a query evaluation plan, said query evaluation plan including a query tree setting a join order for variables in the query;

executing the query evaluation plan, by performing selection, inheritance processing, and joining for expressions and subexpressions in the query,

wherein the executing of the query evaluation plan includes a step of inferencing query results based upon a first set of relational tables in a knowledge representation system and a second set of relational tables in a database system.

5. A knowledge representation system, comprising:

a microprocessor for processing queries;

a display connected to said microprocessor, for displaying input to and output from the microprocessor;

an input device for inputting queries into the microprocessor;

a primary storage medium for storing data which is accessed by said microprocessor;

a secondary storage medium for storing data accessed by the microprocessor;

wherein the primary storage device stores structural assertions which are accessed by said microprocessor in response to a query input from said input device, and wherein said secondary storage medium stores non-structural assertions accessed by said microprocessor, said structural assertions being stored in the primary storage medium in at least one relational table, and wherein the non-structural assertions are stored in the secondary storage medium in a plurality of binary relational tables.

6. A knowledge representation system as recited in claim 5, wherein said primary storage medium is random access memory.

7. A knowledge representation system as recited in claim 6, wherein said secondary storage medium is a mass storage device.

8. A knowledge representation system as recited in claim 5, wherein said microprocessor, said primary storage medium and said second storage medium are configured to create an integrated knowledge base and database system, wherein queries from said input device can be efficiently processed

utilizing a plurality of relational tables for parsing and inferencing query structures.

9. A computer program embodied on a computer readable medium, said computer program controlling a general purpose computer to perform the steps of:

inputting a query to the knowledge representation system;

parsing the query to identify properties, constants, and variables contained therein;

generating a query evaluation plan, said query evaluation plan including a query tree setting a join order for variables in the query;

executing the query evaluation plan, by performing selection, inheritance processing, and joining for expressions and subexpressions in the query,

wherein the executing of the query evaluation plan includes a step of inferencing query results based upon a first set of relational tables in a knowledge representation system and a second set of relational tables in a database system.

10. A knowledge representation system as recited in claim 1, further comprising an input device for inputting queries to the query processing device, said input device including a graphical query generation device for specifying and generating complex queries as a combination of constants, variables, and relationships therebetween.

11. A knowledge representation system as recited in claim 5, wherein said input device includes a graphical query generation device for specifying and generating complex queries as a combination of constants, variables, and relationships therebetween.

12. A knowledge representation system as recited in claim 10, further comprising an application program interface for controlling communication between the graphical query generation device and the knowledge representation module, said application program interface including loading

means for loading selected knowledge information, and a data manipulation device for manipulating selected data.

13. A knowledge representation system as recited in claim 11, further comprising an application program interface for controlling communication between the graphical query generation device and the knowledge representation module, said application program interface including loading means for loading selected knowledge information, and a data manipulation device for manipulating selected data.

14. A knowledge representation system as recited in claim 1, wherein said query processing device includes an inheritance application device for applying an inheritance algorithm to query elements during the processing of the queries input into the knowledge representation system.

15. A method of processing queries as recited in claim 4, wherein said step of executing the query evaluation plan includes a step of applying an inheritance algorithm to the query evaluation plan for appropriate inheritance processing of the query.

16. A knowledge representation system as recited in claim 5, further comprising inheritance processing means connected to the microprocessor, the primary storage medium, and the secondary storage medium, for applying an inheritance algorithm to the queries input into the microprocessor for accessing of the structural assertions and the non-structural assertions.

17. A computer program as recited in claim 9, wherein said step of executing the query evaluation plan performed by the general purpose computer includes a step of applying an inheritance algorithm to the query evaluation plan for appropriate inheritance processing of the query.

18. A knowledge representation system as recited in claim 1, wherein said query processing device includes means for generating a left deep tree query evaluation plan.

19. A knowledge representation system as recited in claim 18, further comprising means for performing a natural join on tuples which satisfy individual subexpressions of the query evaluation plan.

20. A knowledge representation system as recited in claim 19, wherein said query processing device further includes execution means for executing the query evaluation plan.

21. A method of processing queries as recited in claim 4, wherein said generating step comprises the step of generating the query evaluation plan as a left deep tree query evaluation plan.

22. A method of processing queries as recited in claim 21, comprising the further step of performing a natural join on tuples which satisfy individual subexpressions of the query evaluation plan.

23. A knowledge representation system as recited in claim 5, further comprising means for generating a left deep tree query evaluation plan connected to said input device.

24. A knowledge representation system as recited in claim 23, further comprising means for performing a natural join on tuples which satisfy individual subexpressions of the query evaluation plan.

25. A knowledge representation system as recited in claim 24, wherein said query processing device further includes execution means for executing the query evaluation plan.

26. A computer program as recited in claim 9, said computer program controlling the general purpose computer to perform the generating step such that the query evaluation plan is a left deep tree query evaluation plan.

27. A computer program as recited in claim 26, further comprising a step of performing a natural join on tuples which satisfy individual subexpressions of the query evaluation plan.

28. A knowledge representation system as recited in claim 1, wherein said query processing device includes database operator integration means wherein database operators are adapted to be queries to the knowledge representation system.

29. A method of processing queries as recited in claim 4, further comprising a step of adapting database operators to be used during the step of executing the query evaluation plan.

30. A knowledge representation system as recited in claim 5, wherein said query processing device includes database operator integration means wherein database operators are adapted to queries to the knowledge representation system.

31. A computer program as recited in claim 9, wherein said computer program controls the general purpose computer to adapt database operators to be used during the step of executing the query evaluation plan.

32. A knowledge representation system as recited in claim 2, further comprising a primary memory device connected to said database module, said knowledge representation module, and said query processing device, and wherein said relational tables in said database module comprise a plurality of subtables, said system further comprising paging means for paging said subtables to memory during query processing.

33. A knowledge representation system as recited in claim 5, wherein said secondary storage medium also stores structural assertions in at least one relational table therein, said system further comprising paging means for dividing said relational tables in said secondary storage medium into subtables, and paging said subtables to said primary storage medium.

34. A knowledge representation system as recited in claim 1, wherein said query processing device processes queries including predicates with multiple arguments.

35. A method of processing queries as recited in claim 4, wherein said step of inputting a query to the knowledge representation system comprises a step of inputting a query including predicates with multiple arguments.

36. A knowledge representation system as recited in claim 5, wherein said input device inputs queries comprising predicates with multiple arguments.

37. A computer program as recited in claim 9, wherein said computer program controls said general purpose computer such that the query input to the knowledge representation system comprises predicates with multiple arguments.

38. A knowledge representation system as recited in claim 1, wherein said query processing device comprises a single microprocessor responding to a series of commands.

39. A knowledge representation system as recited in claim 1, wherein said query processing device comprises at least two microprocessors, processing commands in parallel.

40. A knowledge representation system as recited in claim 5, wherein said microprocessor comprises at least two central processing units for executing commands in parallel.

41. A knowledge representation system as recited in claim 5, wherein said microprocessor comprises a single central processing unit for processing commands in series.

42. A knowledge representation system as recited in claim 1, wherein said database module comprises a text retrieval system.

43. A knowledge representation system as recited in claim 1, wherein said database module comprises global positioning system data.

44. A knowledge representation system as recited in claim 1, wherein said database module comprises language translation system data.

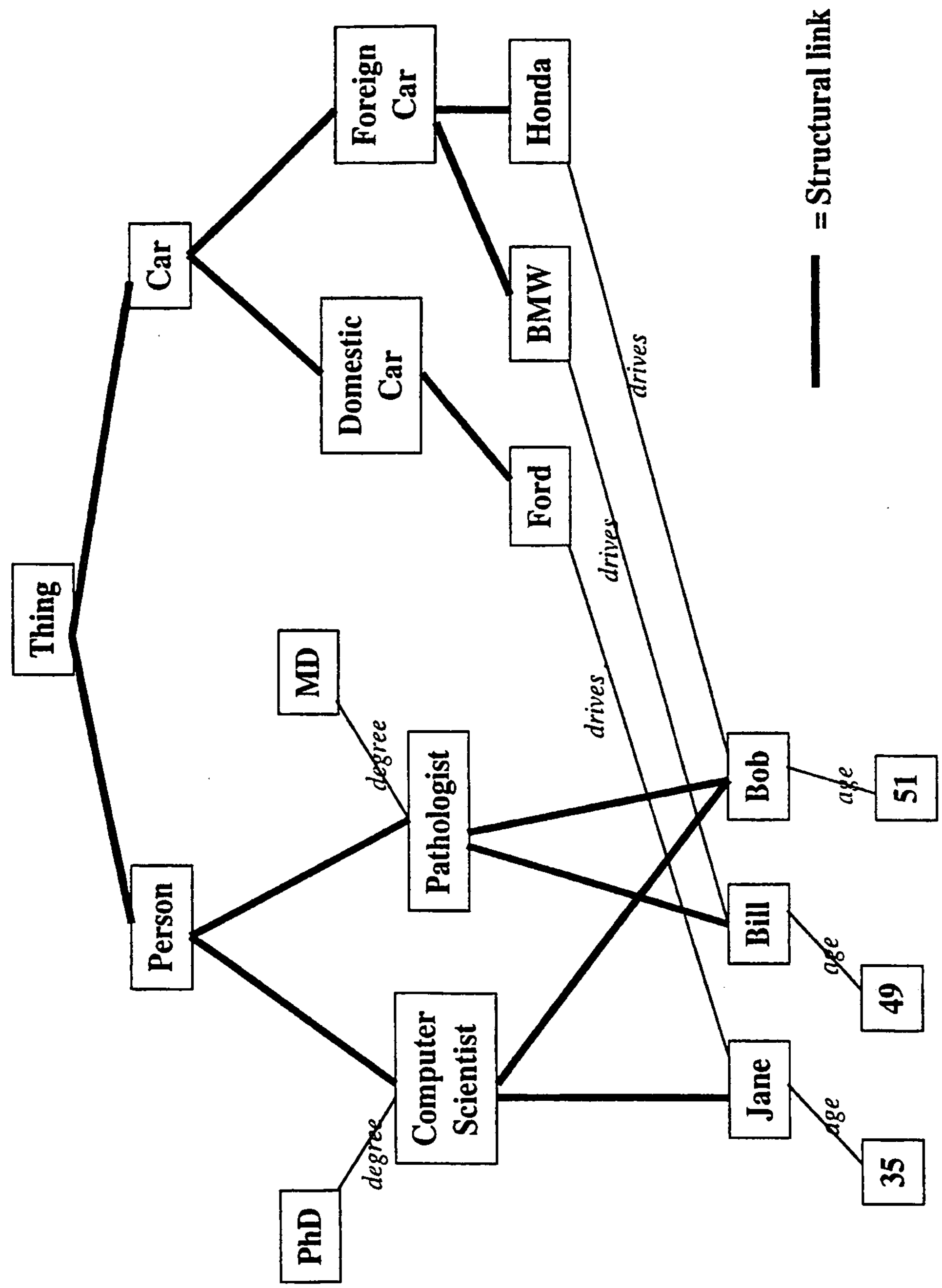

Thing
Car
Person
Domestic Car
Foreign Car
Ford
BMW
Honda
MD
PhD
Pathologist
Computer Scientist
Bob
Bill
Jane
51
49
35
drives
drives
drives
drives
degree
degree
age
age
age
= Structural link

Figure 1

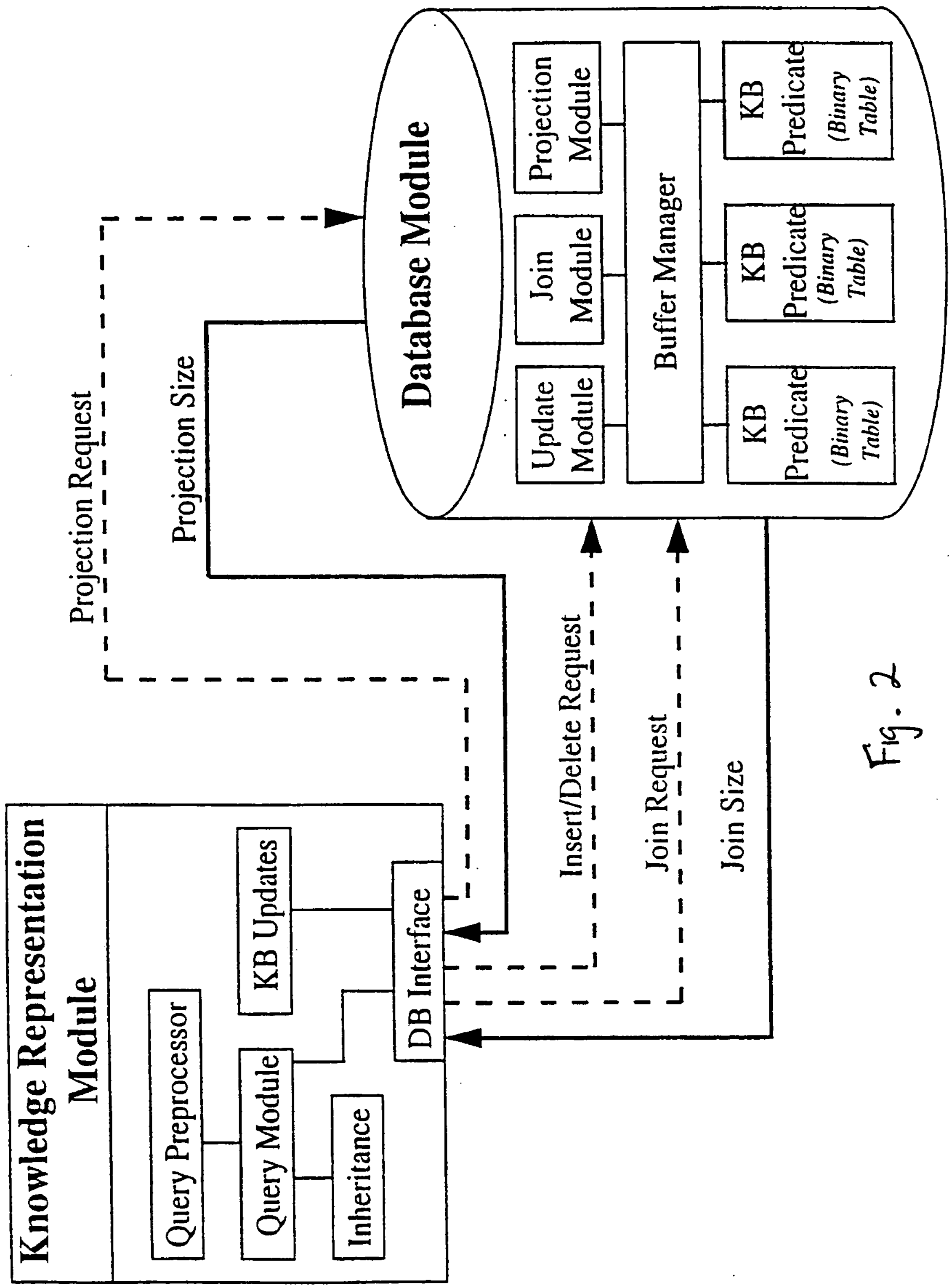

Database Module
Projection Module
Join Module
Update Module
Buffer Manager
KB Predicate (Binary Table)
KB Predicate (Binary Table)
KB Predicate (Binary Table)
Projection Request
Projection Size
Insert/Delete Request
Join Request
Join Size
Knowledge Representation Module
Query Preprocessor
Query Module
Inheritance
KB Updates
DB Interface

Fig. 2

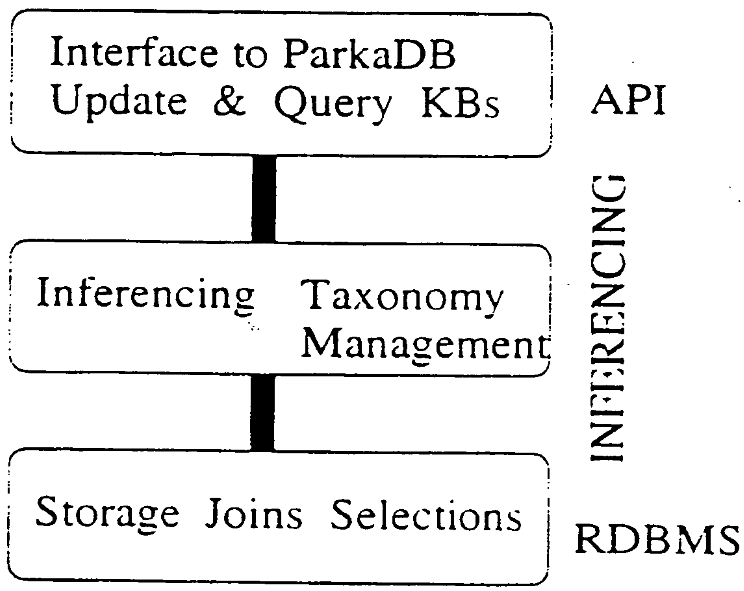

Interface to ParkaDB
Update & Query KBs
API
INFERENCING
Inferencing
Taxonomy Management
Storage Joins Selections
RDBMS

Fig. 3

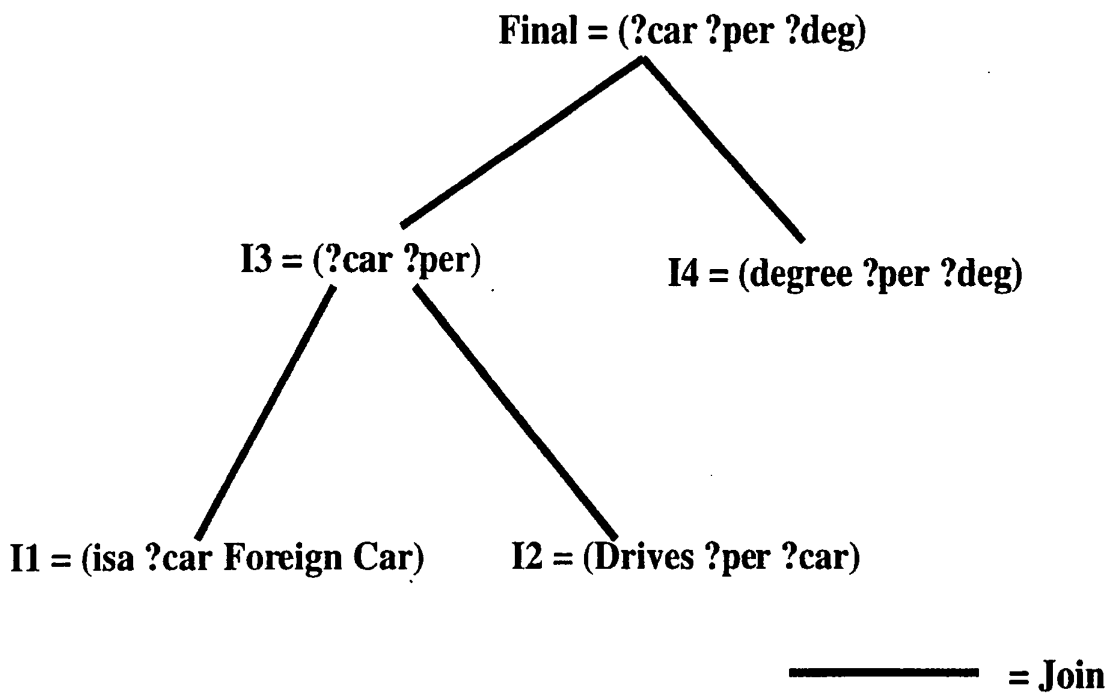

Final = (?car ?per ?deg)
I3 = (?car ?per)
I4 = (degree ?per ?deg)
I1 = (isa ?car Foreign Car)
I2 = (Drives ?per ?car)
= Join

Figure 4

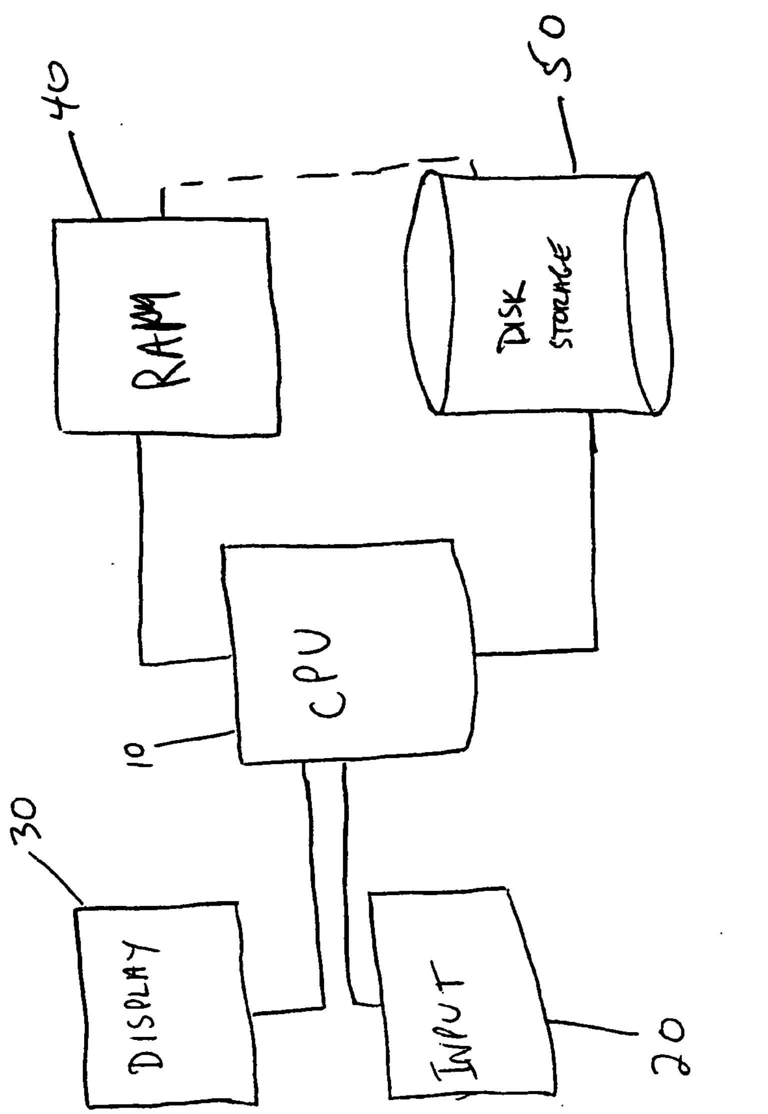


Fig. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/11493

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) : G06F 17/30
US CL : 706/46
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 706/ 46, 706/59, 706/53, 707/101, 707/103, 707/104

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,379,366 A (Noyes) 03 JANUARY 1995 (03.01.95), see entire document.	1-44
A	US 5,630,025 A (Dolby et al.) 13 MAY 1997 (13.05.97).	
A	US 4,964,063 A (Esch) 16 OCTOBER 1990 (16.10.90).	
A	US 5,295,230 A (Kung) 15 MARCH 1994 (15.03.94).	
A	US 5,615,112 A (Liu Sheng et al.) 25 MARCH 1997 (25.03.97).	
A, P	US 5,649,190 A (Sharif-Askary et al.) 15 JULY 1997 (15.07.97).	
A, E	US, 5,805,068 A (Shaw et al.) 08 SEPTEMBER 1998 (08.09.98).	

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
05 OCTOBER 1998	21 OCT 1998

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	WILBERT L. STARKS, JR.
Facsimile No. (703) 305-3230	Telephone No. (703) 308-9700